Enhanced rank-based model for selecting controllers in dynamic and heterogeneous fog environments

Marcus Vinícius Souza Costa*†, Vitor Barbosa Souza†, Xavi Masip-Bruin‡
*Instituto Federal de Educação, Ciência e Tecnologia do Sudeste de Minas Gerais (IF SUDESTE MG), Muriaé-MG, Brazil
†Informatics Department (DPI), Universidade Federal de Viçosa (UFV), Viçosa-MG, Brazil
‡ Advanced Network Architectures Lab (CRAAX), Universitat Politècnica de Catalunya (UPC), Barcelona, Spain
Email: marcus.costa@ifsudestemg.edu.br, vitor.souza@ufv.br, xmasip@ac.upc.edu

***Abstract*—Fog computing is a recent paradigm leveraging available resources at the edge of the network intended to extend the traditional cloud model towards the novel cloud continuum computing model. Recognized the unstoppable growth of highly dynamic and heterogeneous edge devices, as well as the pop up of a large set of diverse and ever more demanding services, the selection of those edge resources best meeting service requirements while also matching the expected QoS guarantees is, with no doubt, a challenging task. This paper presents a rank-based model aimed at both evaluating edge nodes' characteristics and selecting nodes best performing the controller role, whilst simultaneously satisfying the required QoS constraints, coining the so-called Control-as-a-Service concept. To that end, a yet simple prediction strategy, based on Dynamic Branch Prediction is introduced to avoid unnecessary controller exchanges and QoS degradation. In the performed experiments, the proposed method yielded a reduction in the number of exchanges when compared to a solution with no prediction, under different scenarios. Comparing distinct selection strategies, the proposed model presented an improvement in controller availability as well as in relevant controllers' characteristics, such as battery and memory capacity.**

***Index Terms*—Fog computing, Dynamic controller selection, Control-as-a-Service**

I. Introduction

The enormous growth in the number of connected devices observed in the recent years has created several opportunities for innovative services, accommodating people, industry and academia needs and demands. Aligned to this evolution, the Internet of Things (IoT) plays a key role in this scenario by putting together the set of concepts, strategies, services, technologies, and solutions linked to the deployment of a huge variety of devices at the edge of the network. Consequently, a massive volume of data is produced, resulting in large data traffic that is often processed and stored in large data-centers (DCs) located at cloud. However, due to the latency imposed by reaching out to cloud premises, some applications requiring low service response time, such as real-time applications, may experience undesired issues when relying exclusively on cloud computing resources.

To address this issue, approaches aimed at making the most out of the potential of edge devices are gaining momentum, leveraging the utilization of resources available at the edge to increase scalability while reducing the data load forwarded to cloud [1]. Fog computing is designed as an edge computing paradigm intended to extend cloud towards the edge of the network, thus decreasing communication latency, supporting mobility, systems heterogeneity and improving scalability [2]. Fog computing applications may fit diverse scenarios, particularly those with strict demands on real-time data processing, such as dependable services in e-health, or other applications in a diverse set of scenarios, including smart cities, smart vehicles, sensor and actuator networks, and SmartGrids [3].

A key challenge in Fog computing refers to QoS provisioning and is mainly related to the burden imposed on controllers—entities responsible for mapping distinct service requests into the most suitable available resources—in centralized fog control planes [4]. In order to mitigate this problem, authors in [5] introduced a distributed control model, referred to as Control-as-a-Service (CaaS), intended to shift the control decisions to some edge devices serving as controllers. This concept further empowers the network reliability in disaster areas or in other scenarios where static controllers may become unavailable. Considering this distributed control model, edge resources may be (on-demand) assigned to play the controller role, thus, being responsible for both gathering underlying edge resources information and mapping received service requests into those resources best matching the service requirements. In a recent work [6], authors presented a rank-based strategy to select the best resources to play the controller role. Albeit the authors show satisfactory results, the frequency observed on controller exchanges (i.e., changes in the controller nodes allocation) may still be relatively high. This is motivated by the dynamic behavior, inherent to the assessed scenario, what undoubtedly turns into a large variation on the computed rank values (RV). Since a controller exchange takes place each time a candidate node's RV is higher than the current controller's RV, a continuous and non-controlled RV variation may result on two or more devices interchanging controller roles repeatedly in a short-term basis, driving an undesired QoS degradation and network overhead.

Assuming this behavior not to be optimal, this paper proposes an RV forecast strategy, based on predicting the RV variation in order to prevent unnecessary controller exchanges. Therefore, the objective of this work is to propose an approach to select potential controllers for CaaS provisioning, through an accurate RV variation prediction mechanism, considering devices and environments particularities whilst minimizing overhead and QoS degradation. Our main contributions are:

- Proposing a strategy for dynamic allocation of controllers into edge devices in highly volatile and heterogeneous environments.
- Minimizing the frequency of controller exchanges in order to enable efficient CaaS provisioning whilst meeting QoS requirements in real-time applications.

Section II reviews resource selection proposals in distinct scenarios. Section III presents an enhanced rank-based approach for selecting controllers in dynamic and heterogeneous environments, such as Fog. In Section IV, the proposed approach is evaluated. Finally, Section V concludes the paper and gives some directions for possible future work.

II. Related work

The selection of resources to play as centralized controllers in distributed scenarios is a challenging topic in current research. Existing approaches seek to solve different objectives, like energy efficiency, computational resource consumption, reduce latency, and/or network mapping, just to name a few. Some of these works are revisited next.

Arkian *et al.* [7] present a cluster-based approach for resource selection in a vehicular cloud architecture. The presented strategy selects vehicles to perform the role of clusterheads (CH) using fuzzy logic and reinforcement learning. However, authors do not consider the processing and memory capacity of each candidate node during selection. In [8], authors dynamically select stable CHs in VANETs, but, since their work focuses on routing, the selection is performed according to vehicles' positioning, and the heterogeneity of devices is not considered.

Albeit the final purpose of the work presented in [9] is resource discovery in Mobile Cloud Computing (MCC), an appealing clustering-based mechanism is also proposed for CH selection. Similar to the work proposed in this paper, the CH is used as a resource manager for discovering and selecting resources for services execution. However, authors do not consider anticipating controllers exchange to cope with predictable failures.

In [10], authors propose an online secretary framework for dynamic fog network formation in which a given fog node selects the most suitable set of neighboring fog nodes for service offloading. However, no policy for selecting the first node is presented. Kim *et al.* [11] formulate a dynamic market game that performs an economic analysis among the Internet Service Provider (ISP), end service-users, and edge resource owners in a fog environment. In the proposed approach, the ISP is a sort of static controller that mediates service requests through the dynamical employment of edge resources.

III. Enhanced model for controller selection

This section presents an enhanced rank-based model aiming at selecting a set of suitable controllers to provide CaaS in dynamic and heterogeneous environments, such as the one built upon Fog computing.

The next subsection briefly discusses controllers particularities in the envisioned distributed scenario. Subsequently, a strategy for computing candidates' RVs is presented, as well as the criteria deployed to select the first controller. The section ends discussing when controller exchanges should occur.

A. Highlighting controller's particularities

Since the controller is responsible for selecting resources for service execution, rather than executing the service itself, it must gather information regarding resources on the underlying nodes. In addition, the controller election should not consider resources related to specific service execution, such as sensors and actuators. Taking this into consideration, it must be clear that any controller selection mechanism must evaluate service-agnostic characteristics of interest (CI) of each candidate node. In this work, we consider the following CIs, all related to QoS provisioning in terms of controller availability or latency:

- mobility: nodes with high mobility are more likely to migrate to distinct fog domains and relinquish the controller role;
- processing and memory capacity: the utilization of powerful devices for low latency control decisions is a trade-off for allocating them for low delay service execution;
- battery power: low battery may result in controller unavailability due to discharging;
- signal strength: low signal strength can increase connection disruption probability as well as packet loss, thus compromising the QoS.

This paper proposes a weight-based metric that makes use of distinct nodes' CIs and computes their RVs in order to discover potential controllers. In real-world deployments, devices may present high heterogeneity in terms of processing, available memory and mobility, among others. It is with no doubt that these idiosyncrasies must be analyzed when selecting the most suitable resources to play the controller role.

B. RV calculation

Each node j is specified by a set of CIs, denoted by $C_j = \{c_{1j}, c_{2j}, ..., c_{nj}\}$ respectively related to a set of weights $W = \{w_1, w_2, ..., w_n\}$, where n is the number of considered CIs, $0 \leq c_{ij} \leq 100$ and $0 \leq w_i \leq 1$, where $1 \leq i \leq n$. Therefore, the RV for a node j is calculated as:

$$RV_j = \sum_{i=1}^{n} c_{ij} \times w_i \tag{1}$$

As previously mentioned, the CIs considered in this work for RV calculation are mobility, processing power, available memory, battery, and signal strength. The following lines introduce the proposed strategy to compute the value for each CI. It is worth mentioning that a candidate node gives up the election process if any of its CIs is equal to 0 (zero).

- Mobility: The mobility of node j (D_j) contributes so that nodes with a low probability of displacement have higher score values in this metric. Therefore, D_j is given by

  $$D_j = 100 - F_{d_j} \tag{2}$$

  where F_{d_j} is a displacement factor of j obtained through positioning techniques, such as triangulation or GPS, within an interval $\Delta t = t_2 - t_1$. Thus, F_{d_j} is given by

$$F_{d_j} = min\{100, \frac{100}{d_{max}} \sqrt{(x_{t2} - x_{t1})^2 + (y_{t2} - y_{t1})^2}\} \quad (3)$$

where (x_{t1},y_{t1}) and (x_{t2},y_{t2}) are the location coordinates of j at the instant t_1 and t_2, respectively, while d_{max} is a predefined constraint used to normalize the distance traveled by a node regarding the maximum expected displacement of a controller node within an interval Δt.

- Battery: The battery metric (B) indicates the remaining battery in a node, given by the percentage of battery (b). A node must have b greater than a predefined threshold b_{min} to be able to be a controller candidate. To avoid the unavailability of controllers, if a controller has b less than b_{min}, a new selection of controllers must be triggered. Therefore, B_j is given by

$$B_j = \begin{cases} 100 \times b_j & , \text{ if } b_j \geq b_{min} \\ 0 & , \text{ otherwise} \end{cases} \quad (4)$$

- Signal: To assess signal strength (S) of node j, the signal-to-noise ratio (SNR) is normalized according to s_{max}, a predefined base value. Therefore, S_j is given by

$$S_j = min\{100, \frac{100}{s_{max}} \times SNR_j\} \quad (5)$$

- Memory: A controller candidate must have enough available memory (M) to store information about the underlying edge resources. This model classifies the available memory of every node into η possible levels ($\eta \geq 2$). Thus, M_j is given by

$$M_j = \begin{cases} 100 \times \delta^{log_2(\eta-1)-log_2 m_j} & , \text{ if } m_j > 0 \\ 0 & , \text{ otherwise} \end{cases} \quad (6)$$

where m_j ($0 \leq m_j < \eta$) is the memory level of j, and δ ($0 < \delta < 1$) is a constant used to represent the variation of M according to the memory level. For instance, for $\delta = 0.5$, the variation of M is linear since $M = 100 \times 2^{-1(log_2(\eta-1)-log_2 m_j)} = 100 \times m_j/(\eta - 1)$. Setting $\delta > 0.5$ is useful if we intend to achieve higher variations of M as the memory of a node shifts between lower levels, and lower variations when shifting between higher levels. The opposite behavior can be achieved with $\delta < 0.5$.

- Processing: The processing capacity metric (P) is classified through hardware MIPS into ρ distinct levels ($\rho \geq 2$), according to processing capacity parameters. The maximum value (P = 100) is assigned to a node j if its level p_j equals ω ($0 < \omega < \rho$), whilst lower P values are assigned to nodes with higher or lower p_j. Thus, P_j is given by

$$P_j = \begin{cases} 100 \times \phi^{log_2\omega - log_2 p_j} & , \text{ if } 0 < p_j \leq \omega \\ 100 \times \Phi^{log_2(\rho-\omega+1)-log_2(\rho-p_j+1)} & , \text{ if } p_j > \omega \\ 0 & , \text{ if } p_j = 0 \end{cases} \quad (7)$$

where ϕ ($0 < \phi < 1$) and Φ ($0 < \Phi < 1$) are constants used to define the distribution of P_j according to p_j. In other words, they define a decreasing factor of P according to the distance between p_j and ω.

Notice that, in this approach, nodes with processing level $p_j = \omega$ are considered to have enough capacity for the proper execution of controller duties with no QoS loss due to processing overhead. Hence, the employment of nodes with maximum processing capacity as controllers is not required. The rationale behind this approach is to employ resources that comply with the minimum processing requirements for controllers whilst preserving the most powerful resources for the execution of end-user services.

It is worth highlighting that, due to a protocol design decision, the messages exchanged during controller election use 3 bits to represent the memory and processing levels. Therefore, the maximum value of both η and ρ is 8, which we consider enough to classify and differentiate resources in terms of processing and memory capacity in the studied scenario. In our experiments, we set $\eta = 6$, $\rho = 8$, and $\omega = 3$. The use of $\rho > \eta$ reflects need for enough processing classification levels both higher and lower then ω. However, notice that the parameters for memory and processing metrics may be tuned according to the deployed services demands. Finally, the complete equation for the RV of each candidate node j is given by

$$RV_j = (D_j w_1) + (B_j w_2) + (S_j w_3) + (M_j w_4) + (P_j w_5) \quad (8)$$

C. Controller selection process

This subsection details the steps for selecting and eventually exchanging controllers for dynamic CaaS provisioning. The selection of the first node to play as controller follows a decentralized path as described next. The process starts by all candidate nodes calculating their RV values independently, and pruning all candidate nodes with one or more CI values equal to 0. Moreover, in order to avoid increasing the selection delay, the mobility is not considered in the first controller election, since its calculation takes Δt seconds. Hence, the mobility weight is set to 0 (zero).

Then, each remaining node broadcasts its calculated RV and waits for the reception of other candidates' RV. Since this approach deals with dynamic scenarios with varying amount of nodes, after broadcasting its RV, each candidate must set up a timeout interval. When the candidate node receives a message containing an RV, it compares the received RV with its own RV. If its own RV is greater than or equal to the received one, it restarts its timeout timer and keeps waiting for messages from other candidates. In case its RV is lower than the received one, it gives up on being a controller by canceling the scheduled timeout event.

After messages exchange, all but the candidate with the greatest RV give up, canceling their timeout event. Finally, when the timeout event is triggered in the remaining candidate, it becomes the controller and broadcasts its status. The node with greatest RV resets its timeout timer at most *N-1* times before becoming a controller, where *N* is the number of candidates. If an unlikely tie occurs among two or more candidates, a random number is used as a tiebreaker. Thus, after timeout, involved candidates exchange messages containing a 2 bytes random number and the one with the highest number declares itself as controller. If two or more nodes announce themselves as controllers at the end of the selection process, the same tiebreaking process is employed.

Once the elected controller broadcasts its status, the investigation phase starts. Each non-controller node sends a unicast reply message to the controller informing about its characteristics, including the CIs used for the controller selection. Thus, the controller builds an internal database so it can map received requests into the most suitable resources according to the service requirements. It is worth mentioning that extra characteristics may be included apart from the characteristics used for controller selection, but, since they are service-specific, they are out of the scope of this work. In addition, non-controller nodes send their CI at fixed intervals, hereinafter called rounds, enabling the controller to keep an up-to-date database, including mobility data that shall be employed when exchanging controllers.

At the end of each round, the controller calculates the RV of each node, including its own. Since nodes' CIs can change rapidly over time, the RV of each node may vary widely, which might lead to several controller exchanges. Unfortunately, frequent controller exchanges would certainly result in QoS degradation due to frequent messages exchange for the selection itself, followed by the need for building underlying resources database. Consequently, it is a must to design a strategy to reduce such undesired controller exchanges, while, at the same time, guaranteeing that a candidate more suitable to provide long-term CaaS may assume the controller role, thus, minimizing controller unavailability due to predictable failures, such as disruptions resulting from low SNR or lack of battery, further reducing the negative impact on QoS.

Predictive mechanisms can help identify nodes that are rapidly changing RVs, thus preventing them from becoming controllers. Machine learning techniques may be effective for this purpose, but as nodes may have limited memory and processing capacity, other alternatives may be applied to solve this problem. In addition, using complex mechanisms for predicting RV variation of hundreds of candidates simultaneously would certainly introduce a burden that controllers cannot cope with. For this reason, this work proposes to adapt a simple prediction strategy widely used in microprocessors, so-called Dynamic Branch Prediction. In the proposed approach, the controller keeps a Rank Variation History Table (RVHT) with a 2-bit counter for each node, representing 4 distinct levels. Therefore, once the controller receives the CIs of one node and computes its respective RV, it compares the previous RV with the newly calculated one, so it can determine the RVHT level of the node. Each RV increase yields an advance one step closer to the forth level, whilst, each RV decrease results in one step closer to the first level, as illustrated by Fig. 1.

As the controller calculates each node's RV and updates its internal database, it checks if any RV is greater than its own. Additionally, it checks if the node level in RVHT is equal to level four. Otherwise, it might indicate that if any of these nodes become a controller, it is more likely to be replaced quickly. Therefore, if both comparisons (RV and RVHT level) are satisfied, an exchange may occur. In such case, the current controller notifies the node with the greatest RV amongst the ones satisfying both conditions. After receiving the notification, the new controller broadcasts its controller status and the investigation phase restarts. The controller exchange process is described in Algorithm 1. If two or more candidates present the same RV—and considering that they have RVHT level 4—the current controller randomly chooses one among them.

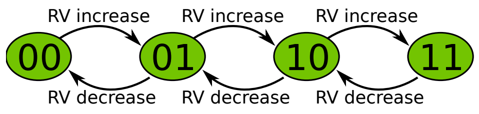

Fig. 1. RVHT with 2-bit saturating counter for predicting RV variation.

Algorithm 1 Controller Exchange

```
1: c: controller node
2: N: set of all nodes
3: t: predetermined round time
4: h: arbitrary node
5: procedure ControllerExchange( )
6:     while $True$ do
7:         $c\ collects\ nodes'\ CI\ for\ t\ seconds$
8:         $c\ calculates\ all\ nodes'\ RV\ including\ its\ own$
9:         $h \leftarrow node\ with\ highest\ RV\ in\ c's\ database$
10:        if ($h's\ RV > c's\ RV$ and $h's\ RVHT\ level = 4$) or
                                   ($c's\ battery\ level < b_{min}$) then
11:            $c\ informs\ h\ that\ it\ is\ the\ new\ controller$
12:            break
13:    $h\ broadcasts\ its\ status\ to\ N$
14: end
```

IV. Performance analysis and comparative study

In order to evaluate the proposed selection strategy, several experiments are carried out to verify the prediction efficiency in reducing unnecessary controller exchanges. Furthermore, the proposed method is also compared with a tailored version of a distinct election strategy found in the literature. In the following subsection, the adapted version used for comparison is described. Later, the performed experiments and the attained results are presented.

A. *Comparative election strategy*

The election strategy tailored for comparison purposes is fully described in [9], where authors present a cluster formation and CH selection strategy for MCC. That work is properly enriched and adapted in this paper to fulfill the requirements imposed by the scenario to be assessed in this work, shifting from ad-hoc to infrastructure mode. For instance, rather than computing the relative mobility through the signal strength variation of each node according to all other nodes, as proposed in the original work, in the infrastructure mode, each node computes the signal strength variation regarding the access point (AP) it is connected to through the received beacon messages. Besides node mobility (Mob), the battery power (B_{power}) is also used to calculate its cluster function ($Clus_{func}$). Hence, $Clus_{func}$ is given by

$$Clus_{func} = w \times Mob + (1 - w)/B_{power} \tag{9}$$

where w is a weight factor, subject to $0 \leq w \leq 1$. In addition, each node broadcasts its $Clus_{func}$ value and the one with lower value is selected as CH. However, the $Clus_{func}$ is not periodically updated by nodes.

Since the usual interval between conventional APs beacon messages is relatively short, the interval between beacons considered for computing signal variation is, at minimum, T_s. Therefore, after listening to the first beacon message, one node must wait at least T_s to consider a new beacon message and compute the signal variation.

It is worth noting that, in this work, we consider one fog domain represented as one single cluster with one CH at a time. Moreover, the comparative strategy, mimicking the original work [9], does not consider the need for reelection when nodes' parameters change over time. Rather than that, a new CH takes place only after a node failure or if a node with a better $Clus_{func}$ value gets into the cluster. On the one hand, this approach reduces the number of controller exchanges. On the other hand, QoS degradation may be expected since, before a failure, a controller may present low throughput and memory overload for several seconds. As a consequence, the controller may relinquish service requests.

B. Scenario description

In the conducted simulations, we deployed fog environments built upon heterogeneous nodes, where the proposed approach is used to dynamically select controllers among available devices as well as the adapted version of CH selection in MCC. The simulations were performed in the OMNeT++ simulator.

The hardware features in edge devices include processing capacity simulation ranging from 1500 to 35000 MIPS and available memory ranging from 150 MB to 3 GB. Features such as SNR, mobility, available memory, and battery availability vary throughout the simulation time. The memory variation is simulated by means of a Probability Density Function (PDF) in inverse gamma distribution with shape parameter $\alpha = 0.5$ and scale parameter $\beta = 10$. A random mobility is implemented through the OMNet++ MassMobility module, while signal power is computed by the OMNet++ simulator. To model the battery consumption, a linear variation is employed. Table I shows the parameter values used for all experiments. It is worth mentioning that s_{max} is defined as 20dB since it is considered a moderate channel quality [12] and a good value for data networks [13]. Since nodes present heterogeneous batteries in terms of both capacity and consumption pattern, b_{min} is set to 10% as a safety margin to prevent controller discharges. Table II, whose values are inferred from [6], shows nodes' memory and processing capacity classification. Notice that some parameters may be tailored according to the specificities of deployed services and scenarios. For instance, processing and memory levels may be tuned in scenarios where the majority of nodes present low processing and memory capacity; services requiring a high updating rate, i.e., short round times, may benefit from a higher value for Δt. The definition of a strategy to set the most suitable values for such parameters is out of the scope of this work. The values for weights w_1 to w_5 are inferred from the adaptive strategy proposed by [14].

TABLE I
EXPERIMENT PARAMETERS

Parameters	Values
Number of nodes	10, 40, 70, 100
Experiments time	3600 rounds
Model thresholds	$b_{min} = 10\%$, $d_{max} = 100m$, $s_{max} = 20db$
Memory and processing constants	$\delta = 0.5, \phi = 0.6, \Phi = 0.7$
Time parameters	Δt = T_s = 5 rounds
Proposed model weights ($w_1 ... w_5$)	$0.2, 0.25, 0.1, 0.25, 0.2$
Comparative model parameter	$w = 0.5$

TABLE II
MEMORY AND PROCESSING CAPACITY CLASSIFICATION LEVELS

Memory		Processing	
Level	Memory (MB)	Level	MIPS
0	less than 128	0	less than 4375
1	from 128 to 255	1	from 4375 to 8750
2	from 256 to 511	2	from 8751 to 13125
3	from 512 to 767	3	from 13126 to 17500
4	from 768 to 1024	4	from 17501 to 21875
5	more than 1024	5	from 21876 to 26250
		6	from 26251 to 30625
		7	more than 30625

C. Results

In order to assess the proposed approach, three distinct aspects are analyzed: number of controller exchanges, controller availability, and characteristics of the used controllers.

The first experiment evaluates the amount of controller exchanges within the simulation time. For the sake of comparison, the proposed method is shown both with and without RVHT prediction. As shown in Fig. 2, the prediction strategy shows a reduction in the number of exchanges for all deployed scenarios. Moreover, when considering all simulations, the average number of controller exchanges is 12.2 when the proposed prediction strategy is employed and 47.9 with no prediction strategy. Hence, the overall reduction is 74.5%. Notice that the adapted version of the CH selection in MCC is not shown in this figure since it focuses on the selection of a CH when none is available, for instance, due to connection or node failure, rather than exchanging CHs.

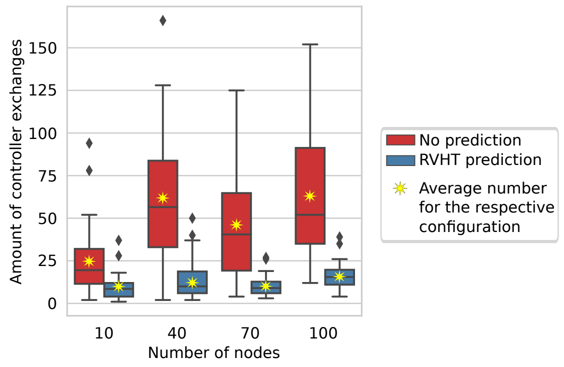

Fig. 2. Number of controller exchanges: boxplot and average.

Fig. 3 illustrates the controller exchanging behavior for both approaches by showing the RV variation of nodes 44, 55, 71 and 89 in an execution slice between rounds 539 and 571, when those nodes are the only ones who assumed the controller role. The two thicker lines highlight the nodes assuming the controller role in each round. Using the proposed prediction mechanism, one single exchange—from node 55 to

89—occurs in round 563. On the other hand, when prediction is not employed, 10 exchanges occur in rounds 541, 542, 550, 551, 555, 556, 561, 566, 567 and 569. When considering the complete execution of this experiment, the chance for exchanging controllers after each round without prediction was 11% and approximately 5% when prediction was employed.

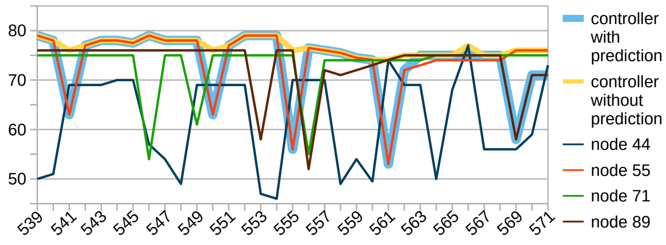

Fig. 3. Nodes' RV variation over time in the proposed strategy.

When selecting resources to play the controller role, a critical requirement is that on-demand selected controllers do not adversely affect service performance. The second experiment compares currently used controllers' characteristics by measuring them in each round. Fig. 4 uses boxplots to show available battery and memory on controllers. Notice that the proposed method has selected controllers with higher battery availability over time, enabling long-term CaaS provisioning. It also shows greater effectiveness than the comparative method for selecting controllers that have more available memory. This means that the controllers selected in the proposed approach are more suitable to cope with large edge resource databases.

Finally, the mechanism used by our proposal to keep up-to-date nodes information enables controllers exchanging before the occurrence of predictable failures, such as lack of battery or connection disruption due to node mobility. In the performed experiments, our proposal achieved a controller availability of 100%, considering only predictable failures, versus 98.54% achieved by the comparative method, where controllers unavailability were observed in 52.6 simulation rounds, in average, out of the total 3600 rounds.

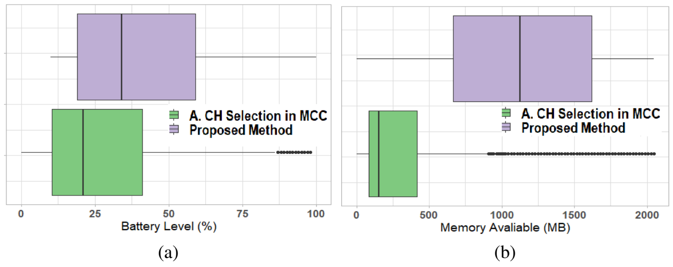

Fig. 4. (a) Battery and (b) Memory availability on controllers.

V. Conclusion

In this paper, we present an enhanced rank-based strategy with RV variation prediction for on-demand selection of controllers. The performed experiments highlight that the proposed approach shows a significant reduction of controllers exchanges. Albeit an adapted approach for CH selection, used as a comparative method, may present fewer controller exchanges, it comes up to present drawbacks such as a higher controller unavailability rate and a notable QoS degradation due to the employment of less suitable resources as controllers. Moreover, the reduced amount of exchanges obtained by the prediction strategy, in comparison to the former strategy with no prediction, does not affect controller availability. As future work, we aim at improving overall resilience by coping with unpredictable controllers failures, and tailoring the model for its deployment in scenarios where ad-hoc communication is demanded. In such cases, it is mandatory to diminish the overhead on the dissemination of nodes' CIs through lightweight protocols, such as gossip.

Acknowledgment

This work is supported by FAPEMIG, CNPq, and CAPES. For the UPC author this work has been supported by the Spanish Ministry of Science, Innovation and Universities and by the European Regional Development Fund (FEDER) under contract RTI2018-094532-B-I00.

References

[1] M. Satyanarayanan, "The emergence of edge computing," *Computer*, vol. 50, pp. 30–39, Jan 2017.

[2] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*, MCC '12, (New York, NY, USA), pp. 13–16, ACM, 2012.

[3] R. Mahmud, K. Ramamohanarao, and R. Buyya, "Application management in fog computing environments: A taxonomy, review and future directions," *ACM Comput. Surv.*, vol. 53, July 2020.

[4] Y. Jiang, Z. Huang, and D. H. K. Tsang, "Challenges and solutions in fog computing orchestration," *IEEE Network*, vol. 32, pp. 122–129, May 2018.

[5] V. B. Souza, A. Gómez, X. Masip-Bruin, E. Marín-Tordera, and J. Garcia, "Towards a fog-to-cloud control topology for QoS-aware end-to-end communication," in *2017 IEEE/ACM 25th International Symposium on Quality of Service (IWQoS)*, pp. 1–5, June 2017.

[6] M. V. S. Costa, V. B. Souza, and S. S. A. Júnior, "Dynamic control-as-a-service provisioning in fog computing," in *2019 International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, pp. 1–6, Sep. 2019.

[7] H. R. Arkian, R. E. Atani, A. Diyanat, and A. Pourkhalili, "A cluster-based vehicular cloud architecture with learning-based resource management," *The Journal of Supercomputing*, vol. 71, no. 4, pp. 1401–1426, 2015.

[8] M. A. Saleem, Z. Shijie, M. U. Sarwar, T. Ahmad, A. Maqbool, C. S. Shivachi, and M. Tariq, "Deep learning-based dynamic stable cluster head selection in vanet," *Journal of Advanced Transportation*, 2021.

[9] P. Athwani and D. P. Vidyarthi, "Resource discovery in mobile cloud computing: A clustering based approach," in *2015 IEEE UP Section Conference on Electrical Computer and Electronics (UPCON)*, pp. 1–6, Dec 2015.

[10] G. Lee, W. Saad, and M. Bennis, "An online secretary framework for fog network formation with minimal latency," in *2017 IEEE International Conference on Communications (ICC)*, pp. 1–6, May 2017.

[11] D. Kim, H. Lee, H. Song, N. Choi, and Y. Yi, "On the economics of fog computing: Inter-play among infrastructure and service providers, users, and edge resource owners," in *2018 IEEE International Conference on Communications (ICC)*, pp. 1–6, May 2018.

[12] D. Lal, A. Manjeshwar, F. Herrmann, E. Uysal-Biyikoglu, and A. Keshavarzian, "Measurement and characterization of link quality metrics in energy constrained wireless sensor networks," in *GLOBECOM '03. IEEE Global Telecommunications Conference (IEEE Cat. No.03CH37489)*, vol. 1, pp. 446–452 Vol.1, 2003.

[13] Cisco Meraki, "Signal-to-noise ratio (SNR) and wireless signal strength." https://documentation.meraki.com/MR/WiFi_Basics_and_Best_Practices/Signal-to-Noise_Ratio_(SNR)_and_Wireless_Signal_Strength, 2020. Accessed: 2021-09-16.

[14] M. V. S. Costa and V. B. Souza, "An adaptive rank-based approach for dynamic controller selection in fog computing," in *Anais do XXXVIII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, (Porto Alegre, RS, Brasil), pp. 113–126, SBC, 2020.